In the Claims

Please amend the claims as follows.

1. (Previously Presented) A method, comprising:

analyzing a dynamic execution trace for a program;

defining at least one stream comprising a sequence of basic blocks in the dynamic execution trace, wherein only a last block in the sequence ends in a branch instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics associated with the at least one stream; and

optimizing the at least one stream based on the metrics.

2. (Original) The method of claim 1, wherein the optimizing comprises encoding the at least one stream as mesocode.

3. (Original) The method of claim 2, wherein the mesocode comprises microinstructions that are ISA-implementation specific.

4. (Original) The method of claim 1, wherein the metrics are selected from the group consisting of a number of instructions within the at least one stream, a number of instructions of each type within the at least one stream, values for particular operands, a coverage for each stream, and a frequency of execution for the at least one stream.

5. (Original) The method of claim 4, further comprising in the case of multiple streams being identified, further comprising identifying streams that are spatially non-contiguous in the dynamic execution trace, but are temporally contiguous.

6. (Original) The method of claim 5, wherein the optimizing comprises encoding the temporally contiguous streams so that they are spatially contiguous in the mesocode.

7. (Original) The method of claim 2, wherein the optimizing comprises including a temporal hint within a basic block of the at least one stream which when executed causes a subsequent block to be prefetched for execution.

8. (Original) The method of claim 7, wherein the subsequent block is a block within the at least one stream.

9. (Original) The method of claim 7, wherein in the case of multiple streams being identified, the subsequent block is a block from another stream.

10. (Previously Presented) A method, comprising:

partitioning a dynamic execution trace for a program into local traces;

analyzing each local trace for streams, each stream comprising a sequence of basic blocks that were sequentially executed, wherein only a last block in the sequence ends in a branch instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics for each stream within a local trace;

for each local trace assigning a locally unique identifier to each unique stream within the local trace, and updating the collected metrics for each unique stream; and

merging stream information from each local stream including assigning a globally unique identifier to each stream that is globally unique across the local traces, and updating the collected metrics for each stream identified by a globally unique identifier.

11. (Original) The method of claim 10, wherein the metrics are selected from the group consisting of a number of instructions within each stream, values for particular operands, a coverage for each stream, a number for each type of instruction within the stream, and a frequency of execution for each stream.

12. (Original) The method of claim 11, further comprising ranking the globally unique streams in accordance with a ranking criterion based on the metrics.

13. (Original) The method of claim 12, further comprising selecting the globally unique streams that have a ranking above a threshold.

14. (Original) The method of claim 13, further comprising forming a control flow graph of program execution wherein each selected globally unique stream defines a node in the control flow graph and each edge between nodes is weighted in accordance with a frequency that the edge was traversed.

15. (Original) The method of claim 14, further comprising pruning edges of the control flow graph that fall below a defined execution frequency.

16. (Original) The method of claim 15, further comprising traversing the pruned control flow graph to extract at least one chain of streams by following the most frequently executed edges from a root of the control flow graph.

17. (Original) The method of claim 16, further comprising optimizing each chain of streams.

18. (Previously Presented) A computer-readable medium, having stored thereon a sequence of instructions which when executed by a computer, cause the computer to perform a method comprising:

analyzing a dynamic execution trace for a program;

defining at least one stream comprising a sequence of basic blocks in the dynamic execution trace, wherein only a last block in the sequence ends in a branch instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics associated with the at least one stream; and

optimizing the at least one stream based on the metrics.

19. (Original) The computer-readable medium of claim 18, wherein the optimizing comprises encoding the at least one stream as mesocode.

20. (Previously Presented) A computer-readable medium, having stored thereon a sequence of instructions which when executed by a computer cause the computer to perform a method comprising:

partitioning a dynamic execution trace for a program into local traces;

analyzing each local trace for streams, each stream comprising a sequence of basic blocks that were sequentially executed, wherein only a last block in the sequence ends in a branch

instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics for each stream within a local trace;

for each local trace assigning a locally unique identifier to each unique stream within the local trace, and updating the collected metrics for each unique stream; and

merging stream information from each local stream including assigning a globally unique identifier to each stream that is globally unique across the local traces, and updating the collected metrics for each stream identified by a globally unique identifier.

21. (Original) The computer-readable medium of claim 20, wherein the metrics are selected from the group consisting of a number of instructions within each stream, values for particular operands, a coverage for each stream, a number for each type of instruction within the stream, and a frequency of execution for each stream.

22. (Previously Presented) A system, comprising:

a processor; and

a memory coupled to the processor, the memory storing instructions which when executed by the processor, cause the processor to perform a method comprising:

analyzing a dynamic execution trace for a program;

defining at least one stream comprising a sequence of basic blocks in the dynamic execution trace, wherein only a last block in the sequence ends in a branch instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics associated with the at least one stream; and

optimizing the at least one stream based on the metrics.

23. (Original) The system of claim 22, wherein the optimizing comprises encoding the at least one stream as mesocode.

24. (Previously Presented) A system, comprising:

a processor; and

a memory coupled to the processor, the memory storing instructions which when executed by the processor, cause the processor to perform a method comprising:

partitioning a dynamic execution trace for a program into local traces;

analyzing each local trace for streams, each stream comprising a sequence of basic blocks that were sequentially executed, wherein only a last block in the sequence ends in a branch instruction, the execution of which causes program flow to branch and end the at least one stream on a taken branch, the remaining basic blocks in each stream each ending in a branch instruction, the execution of which does not cause program flow to branch;

collecting metrics for each stream within a local trace;

for each local trace assigning a locally unique identifier to each unique stream within the local trace, and updating the collected metrics for each unique stream; and

merging stream information from each local stream including assigning a globally unique identifier to each stream that is globally unique across the local traces, and updating the collected metrics for each stream identified by a globally unique identifier.

25. (Original) The system of claim 24, wherein the metrics are selected from the group consisting of a number of instructions within each stream, values for particular operands, a coverage for each stream, a number for each type of instruction within the stream, and a frequency of execution for each stream.